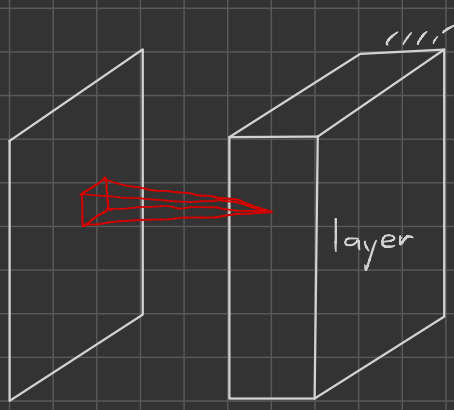


each layer a different filter

space invariant, nonlinear

Convolutional Neural Network

3×3 kernel: $w \in \mathbb{R}^{144}$? 9×16 ??, $b \in \mathbb{R}^{16}$

my intuition:
- different 3×3 kernel for each layer
- offset value for each channel in layer per input data point (16)

$$z_1 = \sigma(y * w + b)$$

activation function; convolution; vector kernel (offset)

Activation Functions

point-wise - function is applied to each element

$$\sigma(y) = \frac{1}{1+e^{y}}$$

$$\sigma(y) = \begin{cases} 0, & y < 0 \\ y, & y \geq 0 \end{cases}$$

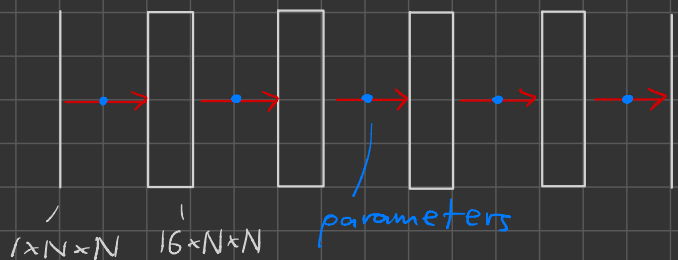
Jeff Hinton
Yoshua Benji
Yon Lacune

rectilinear

$$\sigma(y) = \begin{cases} \alpha y, & y < 0 \\ y, & y \geq 0 \end{cases}$$

$0 < \alpha << 1$

$$\sigma(y) = \frac{e^{y}}{1+e^{y}}$$



15 layers: $160 \times 15 = 2400$ parameters

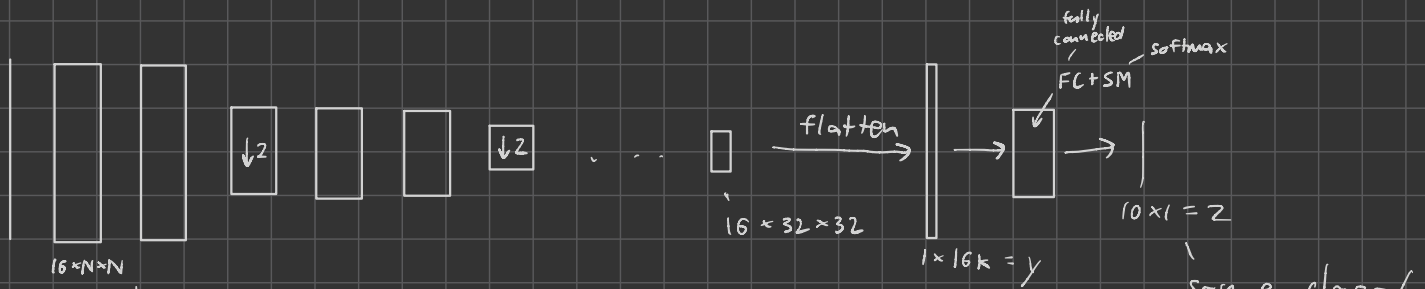
Soft-max activation

tensorflow – "basic" keras
pytorch – "bleeding edge"
"less stable, finnicky"
"more dynamic"

$[\sigma(y)]_i$ $\quad y \in \mathbb{R}^n$ $\quad \frac{e^{y_i}}{\sum_i e^{y_i}} = p_i$ (not pointwise)

$\forall i \; p_i > 0 \; ; \; \sum_i p_i = 1$

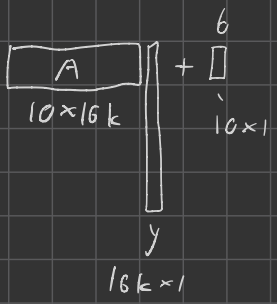
Classification Network



fully connected layer

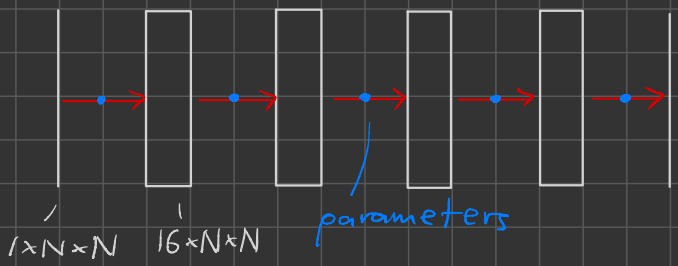
$z = \sigma(Ay + b)$ — 10×1 vector

$A \in \mathbb{R}^{10 \times 16k}$



rube goldberg

Regression Network



Fully Connected Networks/Layers

- more general
- work with classification
- many parameters

Convolutional Layers

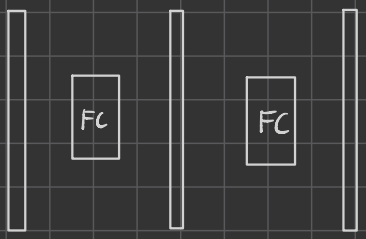
- space invariant
- few parameters
- easier to train

What Changed?

1) intro of convolutional networks
2) stable high performance (GPU-based) software tools for optimization
3) deep networks

Shallow Networks

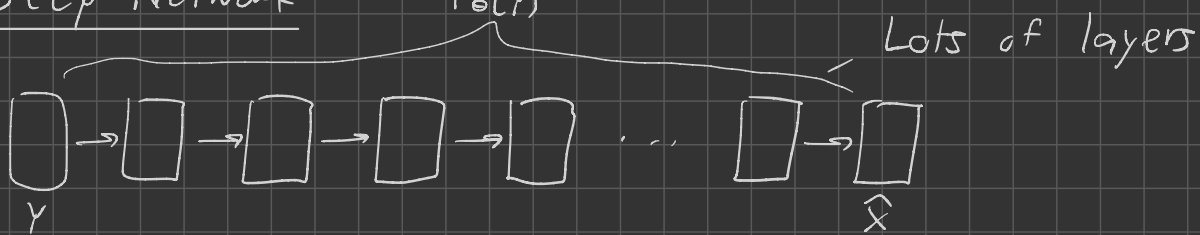
- can model anything: true



$z \in \mathbb{R}^N$

but $N \to \infty$

Deep Network



↳ next question: how to optimize??

$$Loss(\theta) = \|\hat{x} - f_\theta(Y)\|^2$$

- stochastic gradient descent

Backpropogation (chain rule for partial differentiation)

- keep track of chain rule dependencies - stable software can do it